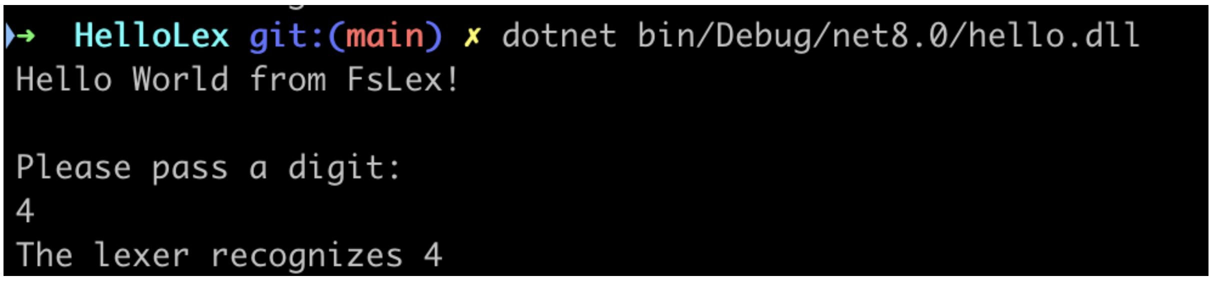
Question 1

The regular expressions involved in Hello.fsl is [‘0’ - ‘9’] that has the semantic value *{ LexBuffer<char>.LexemeString lexbuf }* which can return a string representation of an integer from 0-9.

Theres also a wildcard symbol that catches everything else than the above expression, and throws an error { failwith "Lexer error: illegal symbol" }.

Question 2
After running the command *fslex --unicode hello.fsl* an extra file is generated called: hello.fs. There are 3 states by the automation of the lexer.

Question 3
The program is running:

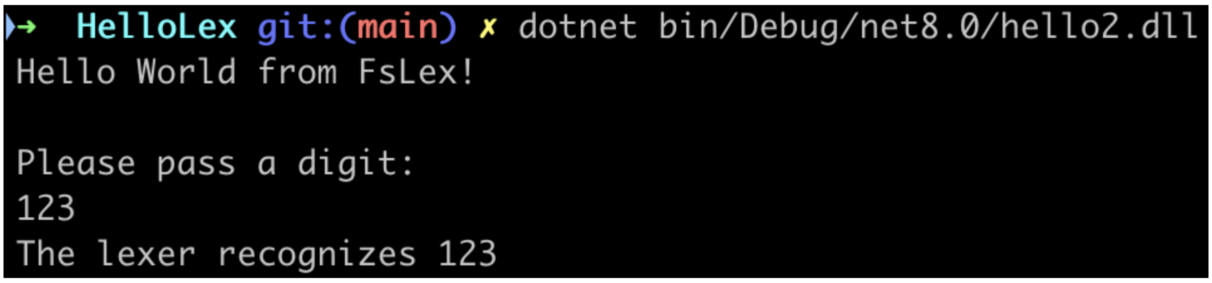


```
➜  HelloLex git:(main) ✗ dotnet bin/Debug/net8.0/hello.dll
Hello World from FsLex!

Please pass a digit:
4
The lexer recognizes 4
```

Question 4
The regular expression was expanded and now looks like this: [‘0’ - ‘9’]*

The program is running:

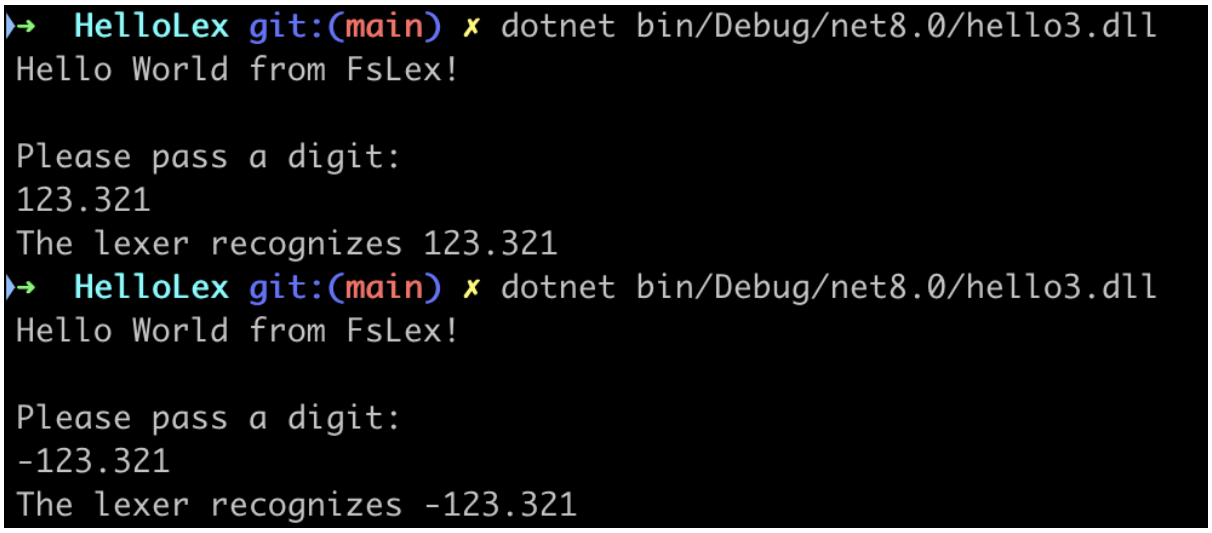


```
➜  HelloLex git:(main) ✗ dotnet bin/Debug/net8.0/hello2.dll
Hello World from FsLex!

Please pass a digit:
123
The lexer recognizes 123
```

Question 5
We expanded the regular expression to recognize this: ['+' '-']?(['0'-'9']*['.'])?['0'-'9']+

The program is running:



```
➜  HelloLex git:(main) ✗ dotnet bin/Debug/net8.0/hello3.dll
Hello World from FsLex!

Please pass a digit:
123.321
The lexer recognizes 123.321
➜  HelloLex git:(main) ✗ dotnet bin/Debug/net8.0/hello3.dll
Hello World from FsLex!

Please pass a digit:
-123.321
The lexer recognizes -123.321
```

Question 6
The lexer will go through the rules that are defined and try to find the longest matching pattern possible.

- For the first example it's fed with the value “34” which is all recognized from the regex.
- The second example it's fed with 34.34 which also is recognized all the way to the end from our regex created from question 5, and therefore returns “34.34”.
- The third example it's fed with “34,34”. For this example, we encounter a symbol we have not defined in our lexer rules and therefore it will match the first “34” from the left side of the “,” without problems. When it reaches the “,” it doesn't recognize the symbol and will therefore be catched by the wildcard pattern “_”, which in our case throws an error *"Lexer error: illegal symbol"*. The rest of the string is then discarded.